Fundamental Analysis of Securities Trading
(II) Data A

Kuan-Lun Wang

National Taiwan University

Preliminary Draft: September 28, 2019

1E-mail: polyphonicared@gmail.com
2Home Page: https://www.csie.ntu.edu.tw/~d06922002/

Short Biodata

Kuan-Lun Wang is a doctoral student majoring in generalized pairs trading. The main goal of his research is to develop an algorithmic trading mechanism based on statistical arbitrage. His areas of expertise include automatic search procedures for model selection, multivariate co-integration approach, and structural change test.
Kuan-Lun Wang’s research interests comprise time series models, simulation modeling, and portfolio choice. The central themes of his application are the study of multivariate pairs trading in real time, search for assets with a long-run equilibrium, and building of riskless portfolios. Much of his current work involves conducting structural change analysis and co-integration test of the finite order vector autoregressive process and estimating the probability of mean reversion. Such methods are important in a variety of applications, including economic indicators and hedging. One such application is index funds being tied to indexes with very low costs and risks.

Source: TWSE (Data E-Shop)
The Data E-Shop of the TWSE provides a way for you to obtain the trading data more easily.


Warning: TWSE (Data E-Shop)
Some data may be only provided in the Traditional Chinese (中文) language page.

https://eshop.twse.com.tw/zh/category/all
**Data E-Shop (2/8)**

### All Products
- **English Version:**
  - End of Day Information
  - Index Information
  - Historical Information

- **中文版：**
  - 盤後資訊
  - 指數資訊
  - 買賣日報表
  - 歷史交易資料

### Source: 買賣日報表
該系統係提供當日各證券商買賣證券統計資料之查詢。


---

**Data E-Shop (3/8)**

### List: End of Day Information vs. 盤後資訊
- **English Version:**
  - End of Day Information
  - SBL

- **中文版：**
  - 三大法人買賣統計
  - 有價證券倉貸
  - 信用交易相關
  - 證券商成交統計
  - 收盤交易統計
  - 公司行為相關
  - 認購售權證
  - 盤後其他資訊

**MAGIC! Why they are so different?**
Anyway, we must search for the needed data.
Daily Quotes (每日收盤行情)

An inexplicable word. We cannot search the explanation for this word by Google. Does anyone know it?

Daily Quotes: https://eshop.twse.com.tw/en/product/detail/ef7b7785e2cb4793baca3644c8a74d4e
每日收盤行情: https://eshop.twse.com.tw/zh/product/detail/cfec9a1470e448ec91bfde006db361e8

Use of Information and Price (用途及價格)

- Internal Use (內部使用): NTD 1,000/Per month
- External Use (外部使用): NTD 1,500/Per month

Cost

If we want to backtest a strategy in $2 + 1 = 3$ years, then the cost is NTD $3 \times 12 \times 1,000 = 36,000$. That is inexpensive!?

So, buy data if we have a not bad strategy.
But, in the class, we had no money.

Are trading data public information? YES.
Public Information: Daily Quotes

In TWSE web, we can find three subpages provide free data in csv-format.

- **English:**

- **Chinese:**

- **Japanese:**

---

**使用條款**

六、智慧財產權的保護
本網站所使用之軟體或程式、網站上所有內容，包括但不限於著作、圖片、檔案、資訊、資料、網站架構、網站畫面的安排、網頁設計，均由臺灣證券交易所或其他權利人依法擁有其智慧財產權。任何人不得逕自使用、修改、重製、公開播送、改作、散布、販售、發行、公開發表、進行還原工程、解編或反向組譯。若您欲引用或轉載前述軟體、程式或網站內容，必須依法先取得臺灣證券交易所或其他權利人之事前書面同意，並請註明資料來源同時確保資料之完整性，不得任意增刪。尊重智慧財產權是您應盡的義務。


臺證實字第1070016756號
WEB, HTML, and Web Browser

- WEB is a computer programming system created by Donald E. Knuth.
- Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications.
- Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages.

https://en.wikipedia.org/wiki/WEB
https://en.wikipedia.org/wiki/HTML

We use web browser to display WEB page in HTML.
A Uniform Resource Locator (URL), colloquially termed a web address, is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.

https://en.wikipedia.org/wiki/URL

**Example: TWSE**

We can use Chrome browser to display a TWSE home page, is identified by a URL http://www.twse.com.tw/en/.

---

**Example: Download a Daily Quotes File (1/2)**

TWSE provides Daily Quotes data in a web page. The page can update CSV-files download link when we send a query. If we click this link in Chrome browser, then browser saves CSV-file.

Example: Download a Daily Quotes File (2/2)

If we open the chrome download page (CTRL+J), then the page show the browser get a CSV-file from a URL http://www.twse.
com.tw/en/exchangeReport/MI_INDEX?response=csv&date=20190226&type=ALL. Moreover, we can change the URL to get other CSV-file. Such as, we change 20190226 to 20190225.

Okay, we only care about this URL.

Example: Download a Security Code List File

TWSE provides security code list in a web page. The page does not provide CSV-file. We need to convert the HTML table to a form file, e.g. CSV-file.

So, not all source are the same format.
But the processes are the same.
The computer execute process:
1. give a URL (we click the link).
2. open the HTML corresponding URL (browser do it).
3. save data (browser do it).

That is, we need to define three function:
1. getURL
2. getData
3. writeDownloadFile
Every HTTP URL conforms to the syntax of a generic URI. The URI generic syntax consists of a hierarchical sequence of five components:

```
URI = scheme:[//authority]path[?query][#fragment]
```

So, we rename `getURL` as `getURI` and define the input and the output are `str`.

That is, the following functions:

- `str: uri=getURI(int: yyyymmdd)`
- `getData`
- `writeDownloadFile`
Comma-Separated Values

In computing, a comma-separated values (CSV) file is a delimited text file that uses a comma to separate values. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas.

https://en.wikipedia.org/wiki/Comma-separated_values

Listing 1: example.csv

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a,b,c</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>d,e,f</td>
<td></td>
</tr>
</tbody>
</table>

Table: example.csv

OK, we know the csv is str and define the intup and the the output are str.
That is, the following functions:

1. `str: uri=getURI(int: yyyymmdd)`
2. `str: data=getData(str: uri)`
3. `NoneType: writeDownloadFile(str: data, str: filename)`

The Web browser save daily quotes data with default file name `MI_INDEX.csv`. If all file has same file name, then we only has the last file. So, we need to define file name for each data.

1. `str: uri=getURI(int: yyyymmdd)`
2. `str: data=getData(str: uri)`
3. `str: filename=getFilename(str: yyyymmdd)`
4. `NoneType: writeDownloadFile(str: data, str: filename)`
Since user maybe want to search a single day or multiple days, the download API is defined as:

```
NoneType: download(int: yyyymmdd[1+])
```

Client (User) uses:

- `str: uri=getURI(int: yyyymmdd)`
- `str: filename=getFilename(str: yyyymmdd)`
- `str: data=getData(str: uri)`
- `NoneType: writeDownloadFile(str: data, str: filename)`
Analysis for URL

- **scheme**: http
- **authority**: www.twse.com.tw
- **path**: /en/exchangeReport/MI_INDEX
- **query**:
  - **response**: csv
  - **date**: 20190227
  - **type**: ALL

Listing 2: getURI.py

```python
def getURI(yyyymmdd):
    scheme = "http"
    authority = "www.twse.com.tw"
    path = "/en/exchangeReport/MI_INDEX"
    query = 
        ["response","csv"],
        ["date",str(20190227)],
        ["type","ALL"]
    uri = scheme+"://"+authority+path+"?"+
    +query[0][0]+"="+query[0][1]+"&" +
    +query[1][0]+"="+query[1][1]+"&"+
    +query[2][0]+"="+query[2][1]
    return uri
```
Listing 3: getURI.py

```python
if __name__=='__main__':
    uri=getURI(20190227)
    print(uri)
```

Listing 4: cmd

```
python getURI.py
response=csv&date=20190227&type=ALL
```

The return of getURI is right but the programming is not nice.

Listing 5: getURI.py

```python
def getURI(yyyymmdd):
    scheme="http"
    authority="www.twse.com.tw"
    path="/en/exchangeReport/MI_INDEX"
    query=[["response","csv"],
           ["date",str(yyyymmdd)],
           ["type","ALL"]]
    uri=scheme+"://"+authority+path+"?"
    for n in range(len(query)):
        uri=uri+query[n][0]+"="+query[n][1]+"&"
    return uri
```
Listing 6: cmd

1. `python getURI.py` [Enter]

The return is not right.

Listing 7: getURI.py

```python
def getURI(yyyymmdd):
    scheme="http"
    authority="www.twse.com.tw"
    path="/en/exchangeReport/MI_INDEX"
    query=[["response","csv"],
           ["date",str(yyyymmdd)],
           ["type","ALL"]]
    uri=scheme+"://"+authority+path+"?"
    for n in range(len(query)):
        uri=uri+query[n][0]+"="+query[n][1]+"&"
    uri=uri[0:-1]
    return uri
```
We can use Web browser to check this URL. Now, we define a function `getData` to get data from this URL.

```
1 python getURI.py
3 response=csv&date=20190227&type=ALL
```

**Listing 8: cmd**

```
1 python getURI.py
3 response=csv&date=20190227&type=ALL
```

We can use Web browser to check this URL. Now, we define a function `getData` to get data from this URL.

The `urllib.request.urlopen` function can be used to open a URL and return a response object. It is often used to fetch data from web pages.

```
Help on function urlopen in module urllib.request:

urlopen(url, data=None, timeout=<object object at 0x0000017A0CB4E250>, *, cafile=None, capath=None, cadefault=False, context=None)

Open the URL url, which can be either a string or a Request object.

*data* must be an object specifying additional data to be sent to the server, or None if no such data is needed. See Request for details.

urllib.request module uses HTTP/1.1 and includes a "Connection:close" header in its HTTP requests.
```

**Listing 9: urllib.request.urlopen**

```
Help on function urlopen in module urllib.request:

urlopen(url, data=None, timeout=<object object at 0x0000017A0CB4E250>, *, cafile=None, capath=None, cadefault=False, context=None)

Open the URL url, which can be either a string or a Request object.

*data* must be an object specifying additional data to be sent to the server, or None if no such data is needed. See Request for details.

urllib.request module uses HTTP/1.1 and includes a "Connection:close" header in its HTTP requests.
```

**More**
Listing 10: getData.py

```python
from urllib.request import urlopen as _urlOpen

def getData(url):
    data=_urlOpen(url)
    return data
```

Listing 11: getData.py

```python
if __name__=='__main__':
    uri='http://www.twse.com.tw/en/exchangeRepository...
    data=getData(uri)
    print(data)
```

The following return is a object.

Listing 12: cmd

```bash
python getData.py
```

We need to read this memory.
The following return is ‘bytes’ class.

```
Listing 14: cmd

1  python getData.py
2  "2019/02/27 Market Summary"
3  "2019/02/27 ..."
```

We need to convert it to `str`. 

```
Listing 14: cmd

1  python getData.py  Enter
2  b'"2019/02/27 Market Summary"
3  \n\n2019/02/27 ...'
```
Listing 15: help of bytes.decode

```
Help on method_descriptor:

decode(self, /, encoding='utf-8', errors='strict')
    Decode the bytes using the codec registered for encoding.

    encoding
    The encoding with which to decode the bytes.
    errors
    The error handling scheme to use for the handling of decoding errors.
    The default is 'strict' meaning that decoding errors raise a UnicodeDecodeError. Other possible values are 'ignore' and 'replace' as well as any other name registered with codecs.register_error that can handle UnicodeDecodeErrors.
```

Listing 16: getData.py

```
from urllib.request import urlopen as _urlOpen

def getData(uri):
    data=_getResponse(uri)
    return data

def _getResponse(uri):
    with _urlOpen(uri) as response:
        data=response.read()
        data=data.decode("utf-8",errors="ignore")
    return data
```
### Listing 17: cmd

```python
python getData.py
```

```
Enter
```

```
"2019/02/27 TWSE Indices"

<table>
<thead>
<tr>
<th>Index</th>
<th>Closing Index</th>
<th>Dir (+/-)</th>
<th>Change</th>
<th>Change (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formosa Index</td>
<td>11,917.42</td>
<td>+</td>
<td>1.22</td>
<td>0.01</td>
</tr>
<tr>
<td>&quot;TAIEX&quot;</td>
<td>10,389.17</td>
<td>-</td>
<td>2.38</td>
<td>-0.02</td>
</tr>
<tr>
<td>&quot;TWSE CG 100 Index&quot;</td>
<td>5,871.19</td>
<td>-</td>
<td>0.10</td>
<td>---</td>
</tr>
<tr>
<td>&quot;FTSE TWSE Taiwan 50 Index&quot;</td>
<td>7,778.88</td>
<td>-</td>
<td>6.12</td>
<td>-</td>
</tr>
<tr>
<td>&quot;FTSE TWSE Taiwan Mid-Cap 100 Index&quot;</td>
<td>7,581.91</td>
<td>+</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>&quot;FTSE TWSE Taiwan Technology Index&quot;</td>
<td>10,031.71</td>
<td>-</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>&quot;FTSE TWSE Taiwan Eight Industries Index&quot;</td>
<td>8,119.24</td>
<td>-</td>
<td>10.12</td>
<td>-0.12</td>
</tr>
<tr>
<td>&quot;FTSE TWSE Taiwan Dividend+ Index&quot;</td>
<td>6,119.75</td>
<td>-</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td>&quot;Taiwan RAFI® EMP 99 Index&quot;</td>
<td>6,157.75</td>
<td>-</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td>&quot;Taiwan Hc100 Index&quot;</td>
<td>5,760.56</td>
<td>+</td>
<td>0.58</td>
<td>0.01</td>
</tr>
<tr>
<td>&quot;Non-Finance Sub-index&quot;</td>
<td>8,736.45</td>
<td>-</td>
<td>10.12</td>
<td>-0.12</td>
</tr>
</tbody>
</table>

...
The following return has carriage return and newline.

Listing 20: return of getData

```
"2019/02/27 TWSE Indices"
"Index","Closing Index","Dir(+/−)","Change","Change(%)"
"Formosa Index","11,917.42","+","1.22","0.01",
"TAIEX","10,389.17","−","2.38","−0.02",
"TWSE CG 100 Index","5,871.19","−","0.10","−−−",
"FTSE TWSE Taiwan 50 Index","7,778.88","−","6.12","−",
"FTSE TWSE Taiwan Mid-Cap 100 Index","7,581.91","+","−",
"FTSE TWSE Taiwan Technology Index","10,031.71","−","−",
```

What’s this?


Carriage Return and Newline

- \r is carriage return, which is what moves where you are typing on the page back to the left (or right if that is your culture).
- \n is new line, which moves your paper up a line.

https://softwareengineering.stackexchange.com/questions/29075/difference-between-n-and-r-n

Listing 21: getData.py

```python
from urllib.request import urlopen as _urlOpen

def getData(uri):
    data=_getResponse(uri)
    data=fixData(data)
    return data

def fixData(data):
    data=data.replace("\r\n","\n")
    return data

def _getResponse(uri):
    with _urlOpen(uri) as response:
        data=response.read()
        data=data.decode("utf-8",errors="ignore")
    return data
```

Kuan-Lun Wang  |  Fundamental Analysis of Securities Trading
Crawler (Download Single Daily Quotes) (21/30)

Listing 22: save as a file

```python
# 1
"2019/02/27 TWSE Indices"
# 2
"Index", "Closing Index", "Dir (+/-)", "Change", "Change (%)"
# 3
"Formosa Index", "11,917.42", "+", "1.22", "0.01",
# 4
"TAIEX", "10,389.17", "-", "2.38", "-0.02",
# 5
"TWSE CG 100 Index", "5,871.19", "-", "0.10", "--",
# 6
"FTSE TWSE Taiwan 50 Index", "7,778.88", "-", "6.12",
# 7
"FTSE TWSE Taiwan Mid-Cap 100 Index", "7,581.91", "+", ",
# 8
"FTSE TWSE Taiwan Technology Index", "10,031.71", "-",
# 9
"FTSE TWSE Taiwan Eight Industries Index", "8,119.24",
# 10
"FTSE TWSE Taiwan Dividend+ Index", "6,119.75", "-",
# 11
"Taiwan RAFI® EMP 99 Index", "6,157.75", "-", "0.95",
# 12
"Taiwan HC 100 Index", "5,760.56", "+", "0.58", "0.01",
# 13
"Non-Finance Sub-index", "8,736.45", "-", "10.12", "-0.12"
# 14
"Non-Electronics Sub-index", "14,642.70", "+", "26.50",
# 15
"Non-Finance Non-Electronics Sub-index", "12,428.38", "+
```

Kuan-Lun Wang  |  Fundamental Analysis of Securities Trading

Crawler (Download Single Daily Quotes) (22/30)

Listing 23: getFilename.py

```python
# 1
def getFilename(yyyymmdd):
# 2    filename='{filename}.{extension}'.format(filename=→
# 3    yyyymmdd, extension='csv')
# 4    return filename
```

Listing 24: writeDownloadFile.py

```python
# 1
def writeDownloadFile(data, filename):
# 2    with open(filename, encoding='utf-8', mode="w") as f
# 3        f.write(data)
```
Now, we can crawl data.

**Listing 25: folder and file**

```python
download.py
__download
|- getURI.py
|- getData.py
|- getFilename.py
|- writeDownloadFile.py
```

**Listing 26: download.py**

```python
from _download.getURI import getURI as getURI
from _download.getData import getData as _getData
from _download.getFilename import getFilename as _getFilename
from _download.writeDownloadFile import writeDownloadFile as _writeDownloadFile

def download(yyyymmdd):
    uri = getURI(yyyymmdd)
    data = _getData(uri)
    filename = _getFilename(yyyymmdd)
    _writeDownloadFile(data, filename)
```
Sometimes, download is crash since the user is banned. If the crash happens, then we should try it again.

**Listing 27: time.sleep**

```python
Help on built-in function sleep in module time:
sleep(...)  # line 3
    sleep(seconds)  # line 4
    Delay execution for a given number of seconds.
    The argument may be a floating point number for subsecond precision.
```

**Listing 28: _download/getData.py**

```python
def getData(uri, tryAgainTime):
    try:
        data = _getData(uri)
    except:
        _pause(tryAgainTime)
        data = getData(uri)
    return data
```
Listing 29: download.py

```python
from time import sleep as _pause

def download(yyyymmdd, tryAgainTime=60):
    uri = _getURI(yyyymmdd)
    data = _getData(uri, tryAgainTime)
    filename = _getFilename(yyyymmdd)
    _writeDownloadFile(data, filename)
```

If the file exists, then we not need to download again.

Listing 30: os.path.isfile

```python
Help on function isfile in module genericpath:

isfile(path)
    Test whether a path is a regular file
```
Listing 31: download.py

```python
from os.path import isfile as _isfile

def download(yyyymmdd, tryAgainTime=60):
    filename = getFilename(yyyymmdd)
    if _isfile(filename):
        return None
    uri = _getURI(yyyymmdd)
    data = _getData(uri, tryAgainTime)
    _writeDownloadFile(data, filename)
```

Listing 32: download.py

```python
def download(yyyymmdd, pauseTime=5, tryAgainTime=60):
    filename = getFilename(yyyymmdd)
    if not _isfile(filename):
        uri = _getURI(yyyymmdd)
        data = _getData(uri, tryAgainTime)
        _writeDownloadFile(data, filename)
    _pause(pauseTime)
```

If we fast crawl two files, then we could be banned.
We rename download as _singleDownload. We use a while-loop to download from yyyyymmddStart to yyyyymmddEnd.

Listing 33: download.py

```python
def download(yyyymmddStart, yyyymmddEnd, 
    pauseTime=5, tryAgainTime=60
):
    yyyymmdd = yyyymmddStart
    onoff = True
    while onoff:
        _singleDownload(yyyymmdd, 
            pauseTime, tryAgainTime
        )
        yyyymmdd = getNextDay(yyyymmdd)
        if yyyymmdd > yyyymmddEnd:
            onoff = False
```

Listing 34: Help of datetime.date

```python
Help on class date in module datetime:

class date(builtins.object)
    date(year, month, day) --> date object
    Methods defined here:
    __add__ (self, value, /)
    Return self+value.
    ...
    strftime(...)
    format -> strftime() style string.
    -- More --
```
Listing 35: Help of datetime.timedelta

```python
Help on class timedelta in module datetime:

class timedelta(builtins.object)
    Difference between two datetime values.
    
    timedelta(days=0, seconds=0, microseconds=0,
    milliseconds=0, minutes=0, hours=0, weeks=0)
    
    All arguments are optional and default to 0.
    Arguments may be integers or floats, and may be
    positive or negative.
-- More --
```

Listing 36: getNextDay

```python
import datetime as _datetime

def getNextDay(yyyymmdd):
    yyyy, mm, dd = splitYYYYMMDD(yyyymmdd)
    today = _datetime.date(yyyy, mm, dd)
    nextDay = today + _datetime.timedelta(days=1)
    nextDay = int(nextDay.strftime("%Y%m%d"))
    return nextDay
```
But the user may want to crawl only one-day data.

**Listing 37: download.py**

```python
def _getYYYYMMDDStartAndEnd(yyyymmddInterval):
    if type(yyyymmddInterval)==list:
        yyyymmddStart=yyyymmddInterval[0]
        yyyymmddEnd=yyyymmddInterval[1]
    else:
        yyyymmddStart=yyyymmddInterval
        yyyymmddEnd=yyyymmddStart
    return yyyymmddStart, yyyymmddEnd
```

**Listing 38: download.py**

```python
def download(yyyymmddInterval, 
              pauseTime=5, tryAgainTime=60):
    yyyymmddStart, yyyymmddEnd=_getYYYYMMDDStartAndEnd(yyyymmddInterval)
    yyyymmdd=yyyymmddStart
    onoff=True
    while onoff:
        _singleDownload(yyyymmdd, 
                        pauseTime, tryAgainTime)
        yyyymmdd=_getNextDay(yyyymmdd)
        if yyyymmdd>yyyymmddEnd:
            onoff=False
```